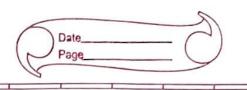
		Name: Deep Pawar
		CWID: A20545137
		5 nb: C5 536 Assignment 6
	3	
		·
<u>(9.</u>	١.	
	aj	
	->	Precondition: n> 0
		Postcondition: x = fac(n)
		where
		$fac(n) = \begin{cases} 1 & \text{if } n=0 \\ n * fac(n-1), & \text{if } n>0 \end{cases}$
	3 2	(n * tac(n-1), it n >0
	b	
	->	loop invariant:
(The loop invariant p is
1		p = 8 = fac (y) > 0 < y < n
		This invariant ensures that
		re holds the factorial of y at each step
		y lies within the valid range of [0, n]
		Loop condition:
		The loop will terminate when y=n meaning the loop
		should run as long as y # n. Hence, the loop condition
		15:
		$B = y \neq n$
	-[
	()	
	—)	Bound expression.
	- 1	A bound expression to must:
		1 Decrease monotonically with each iteration of loop
		@ Eventually reach o, ensuring the loop terminates
	11	



Given the loop starts with y = 0 & increments y by I in each iteraction until y = n the distance from the final value a decreases by 1 in every step. Thus:

Steps to show the working of Bound expression:

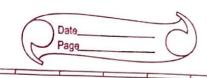
O Initially, t = n - y = n - 0 = n, which is

t = n - y

non- negative

1 In each iteraction, y increments by 1, so t decreases by 1.

3 When y = n, t = n-y = 0, & the loop terminates



- There are many ways to write a program from which one possible full proof outline is as follows: $x := 1 : \{n > 0 \land x = 1 \}$ $\{ y := 0 : \{n > 0 \land x = 1 \}$ { inv p = xe = fac (y) x 0 ≤ y ≤ n } {bd n - y } while y ≠ n do [x = fac (y) 10 & y < n 1 y ≠ n 1 n - y = to] {x * (y + 1) = fac (y + 1) 10 ≤ (y + 1) < n 1 n - (y+1) < to g x := x * (y+1); [x=fac(y+1) 10 < (y+1) < n 1 n - (y+1) < to f y:=y+1 [x = fac (y) 10 ≤ y ≤ n 1 n - y < to } fx = fac(y) AD &y &n Ay = n } x = fac (n) }



(8. Here assume that 0 < i < size (b), 0 < j < size (b) 0 < k < size (b) We need to create a full proof outline for the following minimal proof outline [pfb[if:=b[jf,bfjf:=b[kffb[i]>b[k]f : P1 = {b[i] > b[k]) [b[k]/b[j]] = (if i = j then b[k] else b[i] fi) >

(if k = j then b[k] else b[k] fi) +> (if i= i then b[k] else b[i] fi) > b[k] if i= i then Felse b[i] > b[k] fi →; ≠; ∧ b(i) > b(k) Now, we can find optimized precondition pusing backward assignment & by optimizing P1: $: p = (i \neq j \land b[i] > b[k]) (b[j]/b[i])$ = ; # j ^ b[j] > (if k = i then b[j] else b[k] fi) Hit it is then felse blig > b[k] fi) 1 + 1 × + 1 × b[i] > b[k]

Total State of the last	Q.	4.	
		>	[k < b[k] < b[j] } { [P,] b [b[k]] := b[j]
The second second			$\{b[k] \neq b[j]\}$
Contract designation of the last of the la			Here we have to assume that
distance and deposits of			0 < j < size(b), 0 < k < size(b)
percentant property			Hence
-			the full proof outline for the given minimal proof
A			the full proof outline for the given minimal proof outline is as follows:
Manual de la company			
-			
			all and and



$$P_{L} = (b[k] \neq b[j]) (b[j] / b[b[k]])$$

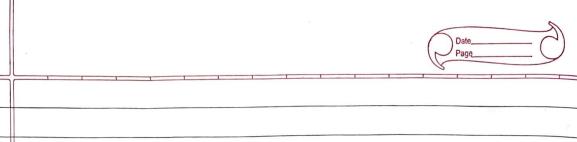
$$= (b[k]) (b[j] / b[b[k]]) \neq (b[j]) (b[j] / b[b[k]])$$

Therefore

Full proof outline is as follows:



Evaluation graph for <5,0> where $S \equiv \{x := 1 \mid x := -1\}; y := y + x$ < <[E|186:=-1]; < | x := 1 || E]; g:=y+80; y:=y+8, σ[x →-1]> <[E||E]; <[EIIE]; y := y + x $\sigma \left(x \mapsto -1 \right)$ y:=y+x, 5[x → 1]> [4 Ho (4)+1]>



Q. 6.

→ Evaluation graph for < W, [x = 0, y = 1, n = 2];

where W = while x < n do [x := x +1 || y := y *2] od

(while x < n do [x := x +1 || y := y *2] od,

[x = 0, y = 1, n = 2];

([x := x +1 || y := y *2]; W, [x = 0, y = 1, n = 2];

([f || y := y *2]; ([x := x +1 || f];

<[E|| y:=y*2];

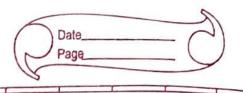
W, {x:=x+1 || E];

W, {x:1, y:4, n:2};
</pre>

< W, {H=2, y=4, n=2}>

<E, {x = 2, y = 4, n = 2}

0	7.	
. 9		·
	al	Are these two threads disjoint?
	>	Yes
		Explanation:
		Here, the two threads are as follows:
		Si = {x = 0 & y := x + 2 {y = 2 }
		52 = { x < 0 } z := 0 { = z > x }
		Two threads are disjoint if they operate on entirely
		separate variables & to not share any dependencies or
		modify each other's values.
		Here,
		Si modifies y, while Si modifies z. Thus, they modify
		distinct variables.
And the second		Neither thread depends on or alters the variable used by the
		other.



Therefore the two threads are disjoint, as they do not share or interact through variables.

b) Do they have disjoint conditions

Explanation:

Two threads have disjoint conditions if their preconditions cannot both be true at the same time.

Here

the precondition of Si is x = 0 the precondition of Si is x < 0

It is impossible for x to simultaneously satisfy x = 0

& x < 0. Therefore, the threads have disjoint conditions as their preconditions cannot overlap.



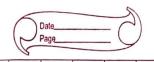
Here the threads written in proof outlines are: Si = [p1] if B1 then [p2] < T1 > else [p3] skip fi fp4? S2 = {q1 } < T2 >; { inv q2 } while B2 do 5 93 4 < T3 > 0 d € 94 4 a) list the interference between treedom checks to decide whether si interferes with Sz. Interference Freedom checks for St interfering with 52 :

We need to determine if the atomic statement Ts in thread SI can interfere with the conditions, invariants, or postconditions in thread 52: The interference freedom checks are: O { p2 1 9 1 } T1 { 91 } : This check ensures that executing TI in the "then" branch of si does not modify variables in a way that invalidates the precondition 91 of 52. @ [p2 1 92 f T1 { 92 f : This ensures that executing TI does not invalidate the invariant 92 during the loop in 52. (3) [p2 193] Ti [93]:

This checks make sure that Ti does not invalidate the condition of used inside the loop body in 52. This ensures that TI does not invalidate the postcondition 94 in 52.



There checks ensure that the atomic statement Ti in Si does not interfere with the preconditions invariants or post-conditions in thread 52. skip interference check: @ Ip3 n q 3 skip q q for the same q: Since skip does not perform any operation this check is trivially satisfied. The invariants 91, 93 & 52 are unaffected. b) List the interference freedom checks to decide whether 52 interferes with 51 Interference Freedom Checks for 52 interfering with 51: We need to consider the atomic statements Tz & T3 in thread 52 & determine if they interfere with the conditions or postconditions in thread 5%: The interference freedom checks are: [1] For T2: () [g 1 ^ p 1 & T 2 { p 1 }; This ensures that executing T2 does not modify variables in a way that invalidates the precondition p1 of si. Q f 91 1 p2 f T2 f p2 f: This check ensures that TZ does not affect the condition p2, which is used after evaluating B1 in SI 3 fq1 x p3 f T2 & p3 f :
This ensures that T2 does not interfere with the assertion p3 in the else" branch of 51.



This check ensures that Tz does not modify any variable that affects the post-condition pt.

[2] For T3:

This ensures that T3, which is executed in the loop body 52, does not interfere with the precondition pl

(6) f q3 A p2 f T3 f p2 f:

This check ensure that T3 does not modify variables

in a way that affects the condition p2 in Si.

This ensures that T3 does not invalidate the assertion

p3 in the "else" branch of 5".

1 f q3 x p4 f T3 &p4 f:

This ensures that T3 does not interfere with the post condition p4 in \$\frac{1}{2} \tag{5}\frac{1}{2}.

This checks ensure that the atomic statements T2 & T3 in S2 do not interfere with the assertion & conditions in S1.